Academia

# Preparación de la integración

Anonymous · 05/11/2026

# Tabla de contenidos

**Preparación de la integración** ..... 3
- ***Cómo funciona la API de LibreDTE*** ..... 4
- ***Tu primera llamada*** ..... 8
- ***El certificado digital*** ..... 12
- ***Folios y CAF*** ..... 17

# Preparación de la integración

Anonymous · 05/11/2026

## Preparación de la integración

Última actualización el 05/11/2026

## Cómo funciona la API de LibreDTE

La forma de las URL, del cuerpo y de las respuestas de los recursos /api/v2/libredte/, la regla del base64 y los errores.

Anonymous · 05/11/2026

# Cómo funciona la API de LibreDTE

La emisión de documentos en API Gateway V2 se hace con los recursos que empiezan con `/api/v2/libredte/`. Todos siguen las mismas reglas, así que si entiendes esta lección vas a poder usar cualquiera de ellos.

## La URL

Cada recurso es una **operación** de la biblioteca de LibreDTE, identificada por cuatro partes:

```
POST
https://app.apigateway.cl/api/v2/libredte/{paquete}/{componente}/{worker}/{operación}
```

Por ejemplo, la operación que construye un documento es:

```
POST https://app.apigateway.cl/api/v2/libredte/billing/document/builder/build
                                                 ──┬─── ──┬─── ──┬─── ──┬─
                                                paquete componente worker operación
```

| Parte | Significado | Valores que usarás en este curso |
|---|---|---|
| `paquete` | Área de la biblioteca | `billing` (facturación), `system` (certificado) |
| `componente` | Tema dentro del área | `document`, `identifier` (folios), `integration` (SII), … |
| `worker` | Quién hace el trabajo | `builder`, `validator`, `dispatcher`, `renderer`, `sii_dte`, … |
| `operación` | Qué se hace | `build`, `validateSchema`, `create`, `render`, … |

La operación se escribe en camelCase, igual que el método de la biblioteca: `validateSchema`, `sendXmlDocument`. El paquete, el componente y el worker se escriben tal como los nombra la biblioteca, por ejemplo `sii_dte`.

Todas las operaciones usan el método **POST**.

## El cuerpo de la solicitud

El cuerpo siempre es un objeto JSON con un único atributo, `parameters`, que contiene los parámetros de la operación con el nombre que declara la documentación:

```
{
    "parameters": {
        "nombre_del_parametro": "valor"
    }
}
```

Por ejemplo, la operación que construye un documento recibe un único parámetro llamado `bag` (la "bolsa" con los datos del documento):

```
{
    "parameters": {
        "bag": {
            "parsedData": { "...": "datos del documento" },
            "caf": "...",
            "certificate": { "...": "certificado digital" }
        }
    }
}
```

## Las cabeceras

```
Authorization: Token <tu_token>
Content-Type: application/json
```

El token es el de tu conexión de API Gateway. A diferencia de las consultas al SII de otros cursos, aquí **no** hay bloque `auth` con RUT y clave: lo que se necesita para hablar con el SII (el certificado digital) va dentro de los parámetros de cada operación.

## La regla del base64

Los datos que no son JSON viajan codificados en **base64**: los XML (del DTE, del sobre, del CAF), los archivos `.p12` y los PDF. Esto evita problemas de caracteres especiales, porque los XML del SII usan la codificación ISO-8859-1 y no UTF-8.

| Dato | Cómo se envía | Cómo vuelve |
|---|---|---|
| Datos del documento | JSON | JSON |
| XML de un DTE, de un sobre o de un CAF | base64 | base64 |

| Dato | Cómo se envía | Cómo vuelve |
|---|---|---|
| Certificado `.p12`/`.pfx` | base64 (con su contraseña aparte) | (no vuelve) |
| Certificado en formato PEM | texto PEM, **sin** base64 | (no vuelve) |
| PDF | (no se envía) | base64 |

Ejemplos para codificar y decodificar en la terminal:

```
# Codificar un archivo en base64, en una sola línea
base64 -i caf.xml | tr -d '\n'

# Decodificar el base64 de una respuesta a un archivo
echo "PD94bWwgdmVyc2lvbj0i..." | base64 -d > dte.xml
```

## La respuesta

Si la operación funciona, la respuesta es un JSON con el resultado dentro de `data`:

```
{
    "data": {
        "...": "resultado de la operación"
    }
}
```

Algunas operaciones no entregan nada (por ejemplo, una validación que no encontró problemas). En ese caso la respuesta es `{"data": null}` con código `200`.

## Los errores

Si algo falla, la respuesta trae el código HTTP y un mensaje en `detail`:

```
{
    "code": 422,
    "detail": "El folio 500 del documento 88888888-8_T033F000000500 no está disponible
en el rango del CAF CAF33D1H100."
}
```

| Código | Qué significa | Qué hacer |
|---|---|---|
| `400` | Falta un parámetro o no tiene el tipo esperado. | Revisa los nombres dentro de `parameters`. |
| `401` | Token inválido o ausente. | Revisa la cabecera `Authorization`. |

| Código | Qué significa | Qué hacer |
|---|---|---|
| `402` | Problema de créditos o de pago de la conexión. | Revisa tu conexión (ver Manejo de errores). |
| `404` | La operación no existe. | Revisa la URL: las cuatro partes deben coincidir con la documentación. |
| `422` | La operación se ejecutó, pero **los datos fueron rechazados**: un RUT inválido, un folio fuera del CAF, un XML mal formado, una firma inválida, etc. | Lee `detail`: explica exactamente qué dato corregir. |
| `500` | Error inesperado. | Reintenta y, si persiste, abre un ticket con la hora y la operación. |
| `503` | El servicio de LibreDTE no está disponible en ese momento. | Reintenta más tarde. |

### El 422 es tu amigo

Casi todos los errores de una integración nueva son `422`, y el mensaje de `detail` dice qué está mal. Muéstralo en tus logs completo: te ahorrará mucho tiempo.

## Dónde ver cada operación

La documentación de la API tiene una ficha por operación, agrupadas como **API LibreDTE - billing.document**, **API LibreDTE - billing.identifier**, etc. Cada ficha indica:

- Los parámetros y un ejemplo del cuerpo.
- Qué entrega y por qué puede rechazar los datos.
- El costo en créditos.

Última actualización el 05/11/2026

## Tu primera llamada

Una primera llamada sin riesgo: listar y obtener los ejemplos de documentos que incluye la API.

Anonymous · 05/11/2026

# Tu primera llamada

Antes de construir documentos, haz una llamada sencilla que no necesita certificado ni folios: pedir los **ejemplos de documentos** que trae la biblioteca. Te sirve para confirmar que el token funciona y, además, te entrega documentos de ejemplo listos para adaptar.

## Listar los ejemplos

```
curl -X POST https://app.apigateway.cl/api/v2/libredte/billing/document/examples/list
\
  -H "Authorization: Token $API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"parameters": {}}'
```

Esta operación no tiene parámetros, por eso `parameters` va vacío. La respuesta es la lista de ejemplos disponibles:

```
{
    "data": [
        {
            "id": "033_factura_afecta/033_001_simple",
            "category": "033_factura_afecta",
            "case": "033_001_simple"
        },
        {
            "id": "052_guia_despacho/052_001_constituye_venta",
            "category": "052_guia_despacho",
            "case": "052_001_constituye_venta"
        }
    ]
}
```

La categoría empieza con el código del tipo de documento: `033` factura, `052` guía de despacho, `061` nota de crédito, etc.

## Obtener un ejemplo

```
curl -X POST https://app.apigateway.cl/api/v2/libredte/billing/document/examples/get \
  -H "Authorization: Token $API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"parameters": {"id": "033_factura_afecta/033_001_simple"}}'
```

Respuesta:


```
{
    "data": {
        "id": "033_factura_afecta/033_001_simple",
        "example": {
            "Encabezado": {
                "IdDoc": {
                    "TipoDTE": 33,
                    "Folio": 1
                },
                "Emisor": {
                    "RUTEmisor": "88888888-8",
                    "RznSoc": "Mi Empresa SpA",
                    "GiroEmis": "Venta de software",
                    "Acteco": 620200,
                    "DirOrigen": "Av. Siempre Viva 742",
                    "CmnaOrigen": "Providencia"
                },
                "Receptor": {
                    "RUTRecep": "60803000-K",
                    "RznSocRecep": "Servicio de Impuestos Internos",
                    "GiroRecep": "Gobierno",
                    "DirRecep": "Santiago",
                    "CmnaRecep": "Santiago"
                }
            },
            "Detalle": [
                {
                    "NmbItem": "Conectores RJ45",
                    "QtyItem": 450,
                    "PrcItem": 70
                }
            ]
        },
        "expected": {
            "Encabezado": {
                "Totales": {
                    "MntNeto": 31500,
                    "IVA": 5985,
                    "MntTotal": 37485
                }
```

```
            }
        }
    }
}
```

### Datos de ejemplo

Los ejemplos de la API traen los datos de un emisor de demostración. En este extracto los reemplazamos por datos ficticios. Cuando emitas, usa siempre los datos de tu empresa.

- `example` son los datos del documento, tal como los enviarás para construirlo.
- `expected` son los totales que debería calcular la API. Fíjate que el ejemplo **no trae totales**: la API los calcula (450 × $70 = $31.500 neto, más 19% de IVA = $5.985).

### Usa los ejemplos como plantilla

Cuando tengas que emitir un caso nuevo (una factura con descuento, una guía de devolución, una nota que corrige montos), busca primero si existe un ejemplo parecido. Es la forma más rápida de saber qué campos lleva.

## Qué revisar si falla

| Respuesta | Causa probable |
|---|---|
| `401` | El token está mal escrito o falta la palabra `Token` antes del token. |
| `402` | La conexión no tiene créditos o tiene un problema de pago. |
| `404` | Hay un error en la URL. |
| `200` con `data` | ¡Todo bien! Sigue con la siguiente lección. |

## En Python

La API es HTTP y JSON, así que funciona con cualquier lenguaje. En Python, con `requests`:

```
import os
import requests

API_URL = 'https://app.apigateway.cl/api/v2/libredte'
HEADERS = {
    'Authorization': f'Token {os.environ["API_TOKEN"]}',
    'Content-Type': 'application/json',
}

def libredte(operacion: str, parameters: dict) -> dict | list | None:
    """Ejecuta una operación de LibreDTE y entrega el contenido de `data`."""
```

```
    response = requests.post(
        f'{API_URL}/{operacion}',
        json={'parameters': parameters},
        headers=HEADERS,
        timeout=120,
    )
    if response.status_code != 200:
        try:
            detalle = response.json().get('detail', response.text)
        except ValueError:  # La respuesta de error no viene en JSON.
            detalle = response.text
        raise RuntimeError(f'{operacion}: HTTP {response.status_code}: {detalle}')
    return response.json()['data']

ejemplos = libredte('billing/document/examples/list', {})
print(len(ejemplos), 'ejemplos disponibles')
```

Usaremos esta función `libredte()` en el resto del curso.

---

Última actualización el 05/11/2026

# El certificado digital

Qué es el certificado digital, en qué formatos enviarlo a la API, cómo verificarlo y cómo protegerlo.

Anonymous · 05/11/2026

# El certificado digital

El **certificado digital** (o firma electrónica) es lo que permite firmar los documentos. Pertenece a una **persona**, no a la empresa: es la persona que el SII autorizó para firmar los documentos de la empresa emisora.

Se usa en tres momentos:

1. Para **firmar el DTE** al construirlo.
2. Para **firmar el sobre** que se envía al SII.
3. Para **autenticarse en el SII** al enviar y al consultar el estado.

**El RUT de la firma no es el de la empresa**

El certificado tiene el RUT de la persona (por ejemplo `11222333-9`), no el de la empresa (`88888888-8`). En el sobre aparece como `RutEnvia`: quien envía el documento en nombre de la empresa. Esa persona debe estar autorizada en el SII para la empresa emisora.

## Formatos que acepta la API

En las operaciones de LibreDTE el certificado va en un objeto `certificate`, en **una** de estas dos formas. En todos los ejemplos del curso usamos PEM, porque es el mismo formato que usan los recursos de consulta al SII.

### Opción 1: certificado y llave privada en PEM (la que usamos en este curso)

```
"certificate": {
    "certificate": "-----BEGIN CERTIFICATE-----\nMIIEcTCCA1mgAwIBAgIBRTAN...\n-----END
CERTIFICATE-----\n",
    "privateKey": "-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhkiG9w0B...\n-----END
PRIVATE KEY-----\n"
}
```

- `certificate`: el certificado público en formato PEM (texto, **sin** base64).
- `privateKey`: la llave privada en formato PEM (texto, **sin** base64).

Normalmente el certificado se entrega como archivo `.p12` o `.pfx`. Para obtener los PEM a partir de ese

archivo, cárgalo con la operación `system/certificate/loader/load`: envía el archivo en base64 y su contraseña, como en la opción 2:

```
curl -X POST https://app.apigateway.cl/api/v2/libredte/system/certificate/loader/load
\
  -H "Authorization: Token $API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "parameters": {
        "certificate": {
            "data": "MIIKkQIBAzCCClcGCSqGSIb3DQEHAaCCCkgEggpEMIIKQDCCBPcGCSqG...",
            "password": "contraseña_del_archivo"
        }
    }
}'
```

La respuesta trae el certificado en `cert` y la llave privada en `pkey`, los dos en PEM:

```
{
    "data": {
        "id": "11222333-9",
        "name": "Daniel Bot",
        "cert": "-----BEGIN CERTIFICATE-----\nMIIEcTCCA1mgAwIBAgIBRTAN...\n-----END
CERTIFICATE-----\n",
        "pkey": "-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhkiG9w0B...\n-----END
PRIVATE KEY-----\n",
        "...": "..."
    }
}
```

Guarda `cert` como `cert.pem` y `pkey` como `key.pem`. Haz esta conversión una sola vez y guarda los PEM como secretos: no la repitas en cada emisión.

#### Los saltos de línea

En JSON, los saltos de línea de un PEM se escriben como `\n`. Si armas el JSON con un lenguaje de programación (leyendo el archivo y serializándolo), esto ocurre solo. Si lo pegas a mano, revisa que no quede todo en una línea sin `\n`.

## Opción 2: archivo `.p12` / `.pfx` en base64

```
"certificate": {
    "data": "MIIKkQIBAzCCClcGCSqGSIb3DQEHAaCCCkgEggpEMIIKQDCCBPcGCSqG...",
    "password": "contraseña_del_archivo"
}
```

- `data`: el archivo `.p12` o `.pfx` completo, codificado en base64 en una sola línea.
- `password`: la contraseña del archivo.

```
base64 -i firma.p12 | tr -d '\n'
```

**No es el mismo formato que en las consultas al SII**

Los recursos de consulta al SII (por ejemplo, los de folios en `/api/v2/sii/dte/caf/...`) reciben el certificado en el bloque `auth.cert` con los campos `cert-data` y `pkey-data` (o `file-data` y `file-pass`). Los recursos de LibreDTE (`/api/v2/libredte/...`) lo reciben en `certificate` con `certificate` y `privateKey` (o `data` y `password`). Son el mismo certificado, con distinto nombre de campos.

| Recurso | Campo | PEM | `.p12` en base64 |
|---|---|---|---|
| `/api/v2/libredte/...` | `certificate` | `certificate` + `privateKey` | `data` + `password` |
| `/api/v2/sii/...` | `auth.cert` | `cert-data` + `pkey-data` | `file-data` + `file-pass` |

## Verificar el certificado

Antes de usarlo para firmar, puedes comprobar que la API lo lee bien y cuándo vence:

```
curl -X POST https://app.apigateway.cl/api/v2/libredte/system/certificate/loader/load
\
  -H "Authorization: Token $API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "parameters": {
        "certificate": {
            "certificate": "-----BEGIN CERTIFICATE-----\nMIIG...\n-----END
CERTIFICATE-----",
            "privateKey": "-----BEGIN PRIVATE KEY-----\nMIIE...\n-----END PRIVATE KEY-
----"
        }
    }
}'
```

La respuesta trae, entre otros datos:

```
{
    "data": {
        "id": "11222333-9",
        "name": "Daniel Bot",
        "email": "daniel.bot@example.com",
        "from": "2026-10-29T11:38:23",
        "to": "2027-10-29T11:38:23",
```

```
        "expirationDays": 364,
        "isActive": true,
        "issuer": "Derafu Test Certificate Authority"
    }
}
```

| Campo | Qué revisar |
|---|---|
| `id` | El RUT de la persona dueña del certificado. |
| `to` / `expirationDays` | Cuándo vence. Un certificado vencido no puede firmar. |
| `isActive` | Debe ser `true`. |

Si falta un dato, la API responde con un error como:

```
{
    "code": 422,
    "detail": "Se requieren los datos del certificado (data y password) o sus llaves
(certificate y privateKey)."
}
```

## Seguridad

El certificado permite firmar documentos tributarios **a nombre de la empresa**. Trátalo como una contraseña:

- Guárdalo cifrado o en un almacén de secretos (variables de entorno del servidor, un vault, el gestor de secretos de tu nube).
- Nunca lo subas a un repositorio ni lo envíes desde el navegador o una app móvil: las llamadas a la API deben salir de **tu servidor**.
- No guardes en tus logs las solicitudes ni las respuestas completas: incluyen el certificado.
- Controla su vencimiento y renuévalo antes de que expire.

## Un certificado de prueba

Mientras sigues el curso puedes construir documentos con un certificado de prueba: la API firmará el XML igual. Lo que no podrás hacer sin un certificado real autorizado es **enviar al SII**, porque el SII rechazará la autenticación.

La API genera uno con esta operación:

```
curl -X POST
https://app.apigateway.cl/api/v2/libredte/billing/trading_parties/mandatario_manager/c
reateFakeCertificate \
  -H "Authorization: Token $API_TOKEN" \
```

```
  -H "Content-Type: application/json" \
  -d '{
    "parameters": {
        "mandatario": {
            "run": "11222333-9",
            "nombre": "Daniel Bot",
            "email": "daniel.bot@example.com"
        }
    }
}'
```

La respuesta (resumida) trae el certificado y su llave privada en PEM, válidos por un año:

```
{
    "data": {
        "id": "11222333-9",
        "name": "Daniel Bot",
        "email": "daniel.bot@example.com",
        "from": "2026-11-04T14:07:03",
        "to": "2027-11-04T14:07:03",
        "issuer": "Derafu Test Certificate Authority",
        "cert": "-----BEGIN CERTIFICATE-----\nMIIE...\n-----END CERTIFICATE-----\n",
        "pkey": "-----BEGIN PRIVATE KEY-----\nMIIE...\n-----END PRIVATE KEY-----\n",
        "...": "..."
    }
}
```

Guarda `cert` como `cert.pem` y `pkey` como `key.pem`, y úsalos como cualquier certificado en PEM: `"certificate": {"certificate": <cert>, "privateKey": <pkey>}`.

---

Última actualización el 05/11/2026

## Folios y CAF

Qué es un CAF, cómo solicitarlo al SII con la API o en el sitio del SII, cómo revisarlo, cómo controlar los folios en tu sistema y cómo practicar con un CAF de prueba.

Anonymous · 05/11/2026

# Folios y CAF

Cada documento tiene un **folio** (su número). No puedes usar cualquier número: el SII te autoriza **rangos de folios** por tipo de documento, y esa autorización llega en un archivo llamado **CAF** (Código de Autorización de Folios).

Por ejemplo, un CAF puede decir: “la empresa `88888888-8` puede emitir facturas (`33`) con los folios del `1` al `100`”.

## Qué contiene un CAF

Es un XML firmado por el SII:

```
<AUTORIZACION>
  <CAF version="1.0">
    <DA>
      <RE>88888888-8</RE>          <!-- RUT del emisor -->
      <RS>Mi Empresa SpA</RS>      <!-- Razón social -->
      <TD>33</TD>                  <!-- Tipo de documento -->
      <RNG><D>1</D><H>100</H></RNG><!-- Rango: desde el 1 hasta el 100 -->
      <FA>2026-10-29</FA>          <!-- Fecha de autorización -->
      <RSAPK>...</RSAPK>           <!-- Llave pública -->
      <IDK>100</IDK>
    </DA>
    <FRMA algoritmo="SHA1withRSA">...</FRMA>   <!-- Firma del SII -->
  </CAF>
  <RSASK>...</RSASK>               <!-- Llave privada para timbrar -->
  <RSAPUBK>...</RSAPUBK>
</AUTORIZACION>
```

La llave privada del CAF (`RSASK`) es la que se usa para **timbrar** cada documento. Por eso el CAF también es un dato sensible: guárdalo con el mismo cuidado que el certificado.

### Reglas de los folios

- Un CAF sirve para **un solo tipo de documento** y **un solo emisor**.
- Un CAF de **certificación** no sirve en **producción**, y al revés.

- Cada folio se usa **una sola vez**. La excepción es un documento **rechazado** por el SII: se considera no emitido, y su folio se vuelve a usar al emitirlo corregido.
- Los CAF de facturas, notas de crédito y notas de débito **vencen a los 6 meses** de autorizados: revisa la fecha de vencimiento al cargarlo. Los de boletas, facturas exentas, guías de despacho y documentos de exportación no vencen.

## Obtener el CAF

Los CAF se solicitan al SII con el certificado digital de una persona autorizada por la empresa. Puedes hacerlo de dos formas, y ambas sirven para **cualquier** CAF, incluido el primero de cada tipo de documento:

| Forma | Cuándo conviene |
|---|---|
| **Con la API** | Para automatizar: tu sistema pide folios cuando se están acabando. |
| **En el sitio del SII** | Para una solicitud puntual, a mano: ambiente de certificación o ambiente de producción. |

### Solicitar un CAF con la API

```
curl -X POST
"https://app.apigateway.cl/api/v2/sii/dte/caf/solicitar/88888888-8/33/100?certificacio
n=1" \
  -H "Authorization: Token $API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "auth": {
        "cert": {
            "cert-data": "-----BEGIN CERTIFICATE-----\nMIIG...\n-----END CERTIFICATE--
---",
            "pkey-data": "-----BEGIN PRIVATE KEY-----\nMIIE...\n-----END PRIVATE KEY--
---"
        }
    }
}'
```

| Parte | Significado |
|---|---|
| `88888888-8` | RUT del emisor |
| `33` | Tipo de documento |
| `100` | Cantidad de folios |
| `certificacion=1` | Ambiente de certificación. Usa `0` (o no lo indiques) para producción. |

La respuesta **no es JSON**: es directamente el XML del CAF, en codificación ISO-8859-1. Guárdalo tal

cual como archivo.

**El SII decide la cantidad**

El SII puede autorizar menos folios de los que pides, según el historial de emisión de la empresa. Revisa siempre el rango real del CAF que recibiste.

### Otros recursos de folios

Todos son `POST`, usan el bloque `auth.cert` y el parámetro `certificacion` igual que la solicitud, y sus URL parten en `/api/v2/sii/dte/caf/`:

| Para qué sirve | Recurso |
|---|---|
| Volver a descargar un CAF ya autorizado, por ejemplo si lo perdiste | `xml/{emisor}/{dte}/{desde}/{hasta}/{fecha}` |
| Ver el historial de CAF solicitados | `solicitudes/{emisor}/{dte}` |
| Ver el estado de un folio en el SII | `estado/{emisor}/{dte}/{folio}` |
| Ver qué folios de un rango están recibidos, anulados o pendientes | `estados/{emisor}/{dte}/{desde}/{hasta}/{estado}` |
| Anular folios que nunca vas a usar | `anular/{emisor}/{dte}/{desde}/{hasta}` |

El detalle de cada uno está en la documentación de la API, en **API DTE - Código de Asignación de Folios (CAF)**.

## Revisar un CAF

Antes de guardar un CAF en tu sistema, cárgalo con la API para leer sus datos. El XML va en base64:

```
curl -X POST
https://app.apigateway.cl/api/v2/libredte/billing/identifier/caf_loader/load \
  -H "Authorization: Token $API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"parameters": {"xml":
"PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iSVNPLTg4NTktMSI/Pgo8QVVU..."}}'
```

Respuesta (resumida):

```
{
    "data": {
        "id": "CAF33D1H100",
        "emisor": {
            "rut": "88888888-8",
```

```
            "razon_social": "Mi Empresa SpA"
        },
        "tipoDocumento": 33,
        "folioDesde": 1,
        "folioHasta": 100,
        "cantidadFolios": 100,
        "fechaAutorizacion": "2026-10-29",
        "fechaVencimiento": "2027-04-27",
        "vigente": true,
        "vence": true
    }
}
```

Guarda en tu base de datos al menos: el emisor, el tipo de documento, `folioDesde`, `folioHasta`, `fechaVencimiento`, el ambiente y el XML del CAF.

## Controlar los folios en tu sistema

La API **no** lleva la cuenta de qué folios usaste: eso es responsabilidad de tu sistema. Una forma simple de hacerlo:

**Tabla de CAF**: un registro por cada CAF recibido.

| emisor | tipo | desde | hasta | vence | ambiente |
|---|---|---|---|---|---|
| 88888888-8 | 33 | 1 | 100 | 2027-04-27 | certificación |

**Tabla de folios usados**: un registro por cada folio que asignas.

| emisor | tipo | folio | estado |
|---|---|---|---|
| 88888888-8 | 33 | 1 | aceptado |
| 88888888-8 | 33 | 2 | enviado |

Para emitir:

1. Busca un CAF vigente del emisor, tipo de documento y ambiente.
2. Toma el siguiente folio libre de ese CAF y **resérvalo** en tu base de datos dentro de una transacción, para que dos emisiones simultáneas nunca usen el mismo folio.
3. Construye el documento con ese folio y ese CAF.
4. Cuando queden pocos folios (por ejemplo, el 20%), solicita un nuevo CAF.

#### El folio debe estar dentro del CAF

Si el folio no está en el rango del CAF, la API responde con un `422`: `El folio 500 del documento 88888888-8_T033F000000500 no está disponible en el rango del CAF CAF33D1H100.`

# Practicar con un CAF de prueba

Mientras aprendes, no necesitas un CAF real. La API puede generar un **CAF ficticio** para cualquier emisor:

```
curl -X POST
https://app.apigateway.cl/api/v2/libredte/billing/identifier/caf_faker/create \
  -H "Authorization: Token $API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "parameters": {
        "emisor": {"rut": "88888888-8", "razon_social": "Mi Empresa SpA"},
        "codigoDocumento": 33,
        "folioDesde": 1,
        "folioHasta": 100
    }
}'
```

La respuesta tiene el mismo formato que `caf_loader/load` y además el XML del CAF en base64 en el campo `xml`. Ese valor es el que usarás como `caf` al construir documentos.

### Un CAF de prueba nunca va al SII

Con un CAF ficticio puedes construir, timbrar, firmar, validar y generar el PDF, pero el SII **rechazará** cualquier documento timbrado con él. Úsalo solo para aprender y para tus pruebas automáticas.

---

Última actualización el 05/11/2026